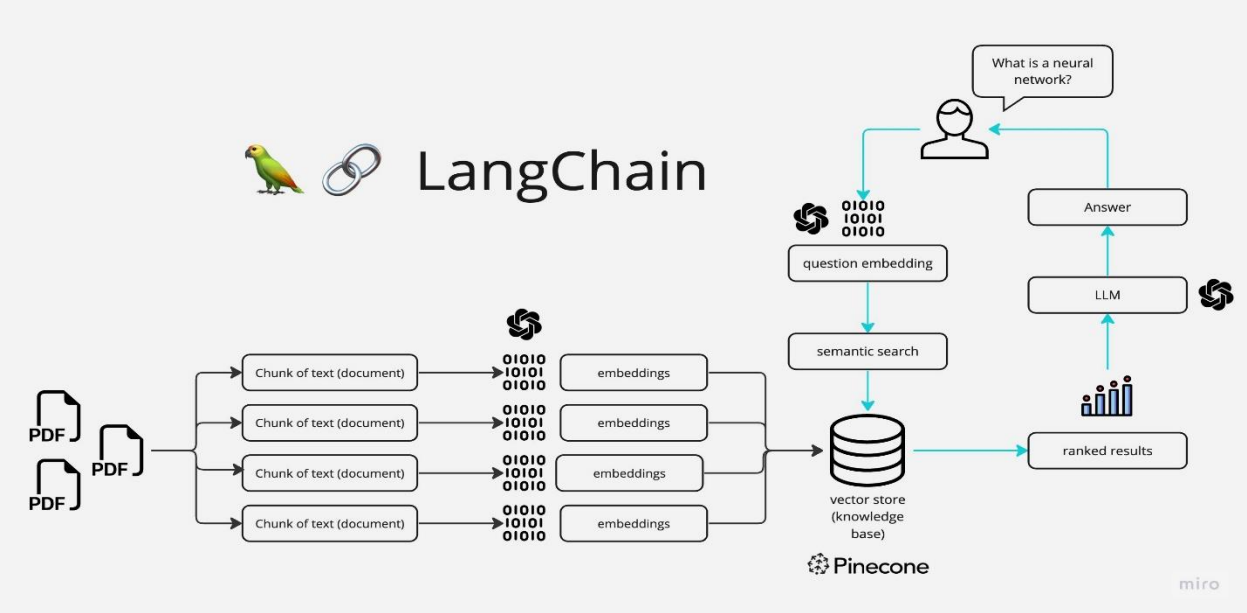
Project Report

Introduction

The MultiPDF Chat App is a Python application that allows you to chat with multiple PDF documents. We can ask questions about the PDFs using natural language processing, and the application will provide relevant responses based on the content of the documents**.** This python scripts utilizes a language model to generate answers to our question. The scripts will only respond to questions related to the loaded PDFs.

How It Work



The python scripts follow these steps:

1. **PDF Loading**: The app reads multiple PDF documents and extracts their text content.
2. **Text Chunking**: The extracted text is divided into smaller chunks that can be processed effectively.
3. **Language Model**: The application utilizes a language model to generate vector representations (embeddings) of the text chunks.
4. **Similarity Matching**: When you ask a question, the app compares it with the text chunks and identifies the most semantically similar ones.
5. **Response Generation**: The selected chunks are passed to the language model, which generates a response based on the relevant content of the PDFs.

Code

```
import streamlit as st
from dotenv import load_dotenv
from PyPDF2 import PdfReader  # PdfFileReader
from langchain.text_splitter import CharacterTextSplitter
from langchain.embeddings import OpenAIEmbeddings, HuggingFaceInstructEmbeddings
from langchain.vectorstores import FAISS
from langchain.chat_models import ChatOpenAI
from langchain.memory import ConversationBufferMemory
from langchain.chains import ConversationalRetrievalChain
from langchain.llms import HuggingFaceHub
from htmlTemplates import css, bot_template, user_template


def get_pdf_text(pdf_docs):
    text = ""
    for pdf in pdf_docs:
        pdf_reader =PdfReader(pdf)
        for page in pdf_reader.pages:
            text += page.extract_text()
    return text
def get_text_chunks(text):
    text_splitter = CharacterTextSplitter(
        separator="\n",
        chunk_size=1000,
        chunk_overlap=200,
        length_function=len
    )
    chunks = text_splitter.split_text(text)
    return chunks
def get_vectorstore(text_chunks):
```

```
    embeddings = HuggingFaceInstructEmbeddings(model_name="hkunlp/instructor-xl")
    vectorstore = FAISS.from_texts(texts=text_chunks, embedding=embeddings)
    return vectorstore
def get_conversation_chain(vectorstore):
    llm = HuggingFaceHub(repo_id="google/flan-t5-xxl", model_kwargs={"temperature":0.5,
"max_length":512})
    memory = ConversationBufferMemory(memory_key='chat_history', return_messages=True)
    conversation_chain = ConversationalRetrievalChain.from_llm(
        llm=llm,
        retriever=vectorstore.as_retriever(),
        memory=memory
    )
    return conversation_chain
def handle_user_qsn(user_quesn):
    response=st.session_state.conversation({'question': user_quesn})
    # st.write(response)
    st.session_state.chat_history=response["chat_history"]

    for i, message in enumerate( st.session_state.chat_history):
        if i%2==0:
            st.write(user_template.replace("{{MSG}}",message.content),unsafe_allow_html=True)

        else:
            st.write(bot_template.replace("{{MSG}}",message.content),unsafe_allow_html=True)

def main():
    load_dotenv()
    st.set_page_config(page_title="Chat with multiple PDFs",page_icon=":books:")
    st.write(css,unsafe_allow_html=True)
```

```
    if "conversation" not in st.session_state:
        st.session_state.conversation=None

    if "chat_history" not in st.session_state:
        st.session_state.chat_history=None

    st.header("Conversation with multiple PDFs :books:")

    user_quesn=st.text_input("Ask a question about your documents:")
    if user_quesn:
        handle_user_qsn(user_quesn)
    with st.sidebar:
        st.subheader("Your documents")
        pdf_docs = st.file_uploader(
            "Upload your PDFs here and click on 'Process'", accept_multiple_files=True)
        if st.button("Process"):
            with st.spinner("Processing"):
                raw_text = get_pdf_text(pdf_docs)
                text_chunks = get_text_chunks(raw_text)
                # st.write(text_chunks)
                vectorstore = get_vectorstore(text_chunks)
                st.session_state.conversation=get_conversation_chain(vectorstore)
if __name__ == '__main__':
    main()
```